Relatório Técnico

Aprendizagem de Padrões de Projeto em Orientação a Objetos: o Sistema Lesoop

H. Gandra
C. Lima

NCE -17/02

Núcleo de Computação Eletrônica

Universidade Federal do Rio de Janeiro

APRENDIZAGEM DE PADRÕES DE PROJETO EM ORIENTAÇÃO A OBJETOS: O SISTEMA LESOOP

Comissão Técnica: Informática

Henrique GANDRA
IBCCF/IM/NCE/UFRJ
Rua Siqueira Campos, 238 / 705, CEP: 22031-070 – Copacabana,
Rio de Janeiro, RJ, Tel: 55 21 2548-5878
hgandra@ufrj.br

Cabral LIMA
DCC/IM/UFRJ
Av. Brigadeiro Trompowsky s/n, Cidade Universitária / Ilha do Fundão,
CEP: 20001-970 Rio de Janeiro, RJ, Tel: 21 2598-3168/ Fax: 21 2598-3156
clima@dcc.ufrj.br

RESUMO

Desde o início da década passada a comunidade de software tem reunido esforços em torno da reengenharia e do reuso de software. Os padrões de projeto em sistemas orientados a objeto vêm sendo considerados como uma das melhores estratégias no aprimoramento de projetos de software baseado na idéia de soluções reutilizáveis. Um padrão de projeto direciona-se a um problema de *design* recorrente que acontece em situações específicas e apresenta uma solução abstrata para esse problema. No entanto, sua aplicação não é simples (posto o custo de complexidade): eles descrevem o problema, a solução (quando aplicável) e suas conseqüências. Faz-se necessário, portanto, que o projetista aprenda a usar corretamente esses padrões. Estamos atualmente desenvolvendo um modelo de plataforma *peer-to-peer*, flexível, destinada ao ensino a distância apoiado por computador, onde é possível o acoplamento de sistemas tutoriais inteligentes. No presente trabalho, descrevemos um sistema tutorial destinado à aprendizagem de padrões de projeto em sistemas orientados a objeto (*design patterns*), de aplicação presencial ou remota. Este sistema, chamado LeSOOP (*Learning System of Object-Oriented Patterns*) utiliza um processo de diagnóstico contextual e uniforme visando auxiliar os projetistas de software na correta aplicação dos padrões de projeto em sistemas legados.

Palavras-chaves: Padrões de Projeto, Engenharia de Software; Ensino a Distância; Sistemas Tutoriais.

1. INTRODUÇÃO

A aprendizagem humana é considerada por muitos autores com um processo longo, lento e complexo (Michalski et al, 1985) por envolver processos cognitivos ainda não totalmente compreendidos. Em aprendizagem a distância, a complexidade é ainda maior devido à diferença de espaço e de tempo entre os processos de ensino e aprendizagem. O ensino a distância, apoiado por computador, por utilizar novas tecnologias de informação e de comunicação, possui uma complexidade

adicional tanto para os alunos quanto para os professores, que nem sempre possuem a capacitação tecnológica que poderia facilitar o uso dessas tecnologias. As ferramentas tecnológicas desenvolvidas para o ensino a distância, apoiado por computador, são usualmente projetadas para funcionar em plataformas educacionais construídas para agrupá-las. Geralmente o uso dessas plataformas tem mostrado que a ênfase está mais nos aspectos tecnológicos do que nos aspectos cognitivos ou nas estratégias de ensino. Além do mais, os usuários dessas plataformas nunca devem sentir a sensação de estarem perdidos, devem ter sempre o controle da interação com a aplicação, a qual deve ser dotada de mecanismos que a protejam de possíveis utilizações erradas por parte dos usuários. Preocupados com isso, pesquisadores têm focado o desenvolvimento de plataformas mais elaboradas, com interfaces amigáveis e de uso fácil e intuitivo e que possam guiar os usuários (alunos e professores) em suas tarefas de ensino e aprendizagem, utilizando técnicas de diagnóstico contextual, de inteligência artificial entre outras.

Neste artigo, primeiramente destacamos a importância dos padrões de projetos em sistemas orientados a objetos para a elaboração de softwares reutilizáveis, que é um objetivo cada vez mais perseguido pela comunidade de engenharia de software. Em seguida apresentamos o LeSOOP, um sistema computacional destinado a dar suporte à aprendizagem presencial ou a distância de padrões de projeto em sistemas orientados a objeto. Este sistema utiliza processos uniformes e contextuais de diagnóstico com o objetivo de auxiliar os projetistas de software na aplicação correta dos padrões de projeto. O LeSOOP será acoplado ao sistema Acádia (Correa e Martins, 2002) – um ambiente de ensino-aprendizagem com interface gráfica amigável - e à plataforma Dedalus (Paiva e Faissal, 2002), uma plataforma que usa a tecnologia "peer-to-peer" e recursos compartilhados, com as características desejáveis citadas acima.

2. PADRÕES DE PROJETO

O arquiteto Chistopher Alexander definiu formalmente pela primeira vez o conceito de padrão: "Cada padrão descreve um problema que ocorre sucessivamente em nosso meio ambiente e então propõe a essência da solução, as diretrizes básicas dessa solução, de forma que você possa reutilizar essa solução um milhão de vezes, sem, no entanto, implementá-la duas vezes da mesma maneira" (Alexander, 1977).

Esta afirmação referia-se a padrões de projetos em construção de prédios e cidades, mas ela se aplica, *ab ovo*, aos padrões de projeto de sistemas orientados a objetos. As soluções para a área de engenharia de *software* estão expressas referindo-se a objetos e interfaces em substituição a paredes e portas, mas as diretivas para resolver um determinado problema, em um determinado contexto, como nos projetos de arquitetura, poderão estar descritas em um padrão de projeto, se for o caso desse problema específico.

A definição de (Gamma et al, 1995) para padrões de projeto é: "Os padrões de projeto são descrições de objetos e classes que se comunicam entre si desenvolvidos sob medida para resolver um problema geral de projeto em um contexto particular". Ainda por (Gamma et al, 1995) um padrão de projeto encapsula estratégias reutilizáveis de interações entre objetos e as descreve como uma possível solução de um problema dentro de um determinado contexto: "cada padrão de projeto focaliza um problema ou tópico particular de projeto orientado a objeto.

Ele descreve quando pode ser aplicado, se ele pode ser aplicado em função de outras restrições de projeto e as conseqüências, custos e benefícios de sua utilização".

Os estudos e definições dos padrões de projeto "nasceram basicamente no início da década de 80, quando o *Smalltalk* era a mais conhecida linguagem de programação orientada a objeto. Naquela época, a linguagem C++ ainda estava em pleno desenvolvimento" (Cooper, 1998) e a programação estruturada era a via clássica e normal de ampla aplicação para a imensa produção de *softwares*, sobretudo os chamados "softwares comerciais". A programação orientada a objetos era utilizada somente por um número restrito de pesquisadores, tendo como sua base de desenvolvimento a *Xerox Palo Alto Research Center*. O *Smalltalk*, por exemplo, foi desenvolvido com o objetivo inicial de ser uma linguagem de programação que se adaptasse à maneira de pensar dos seres humanos e à forma com a qual compreendemos o mundo. A idéia de base é que esta linguagem se tornasse tão natural que pudesse ser amplamente utilizada no ensino de programação para crianças. Uma virtude notória deste enfoque é o fato de poder-se pensar todas as coisas como objetos que podem ser classificados.

As estruturas de programação (*frameworks*) eram então bastante usadas pela comunidade de engenharia de software, que utilizava a programação estruturada. *Frameworks* podem ser vistos como estruturas de programas que são repetidas diversas vezes em muitos sistemas. Os pesquisadores em orientação a objetos também lançaram mão desta idéia e começaram a identificar e fazer uso efetivo das estruturas de programação.

Um dos *frameworks* mais utilizados era o **Modelo/Vista/Controlador** (*Model/View/Controller*) – MVC – para *Smalltalk* (Krasner e Pope, 1988), que dividia a interface do usuário em três partes (Cooper, 1998):

- O objeto **Modelo**: contém a porção computacional do programa.
- O objeto **Vista**: que é a interface de representação visual do modelo.
- O objeto **Controlador**: responsável pela interação entre o usuário e a vista.

Neste caso, cada um dos aspectos do programa é um objeto separado e cada um desses objetos tem suas próprias regras de gerenciamento dos seus dados.

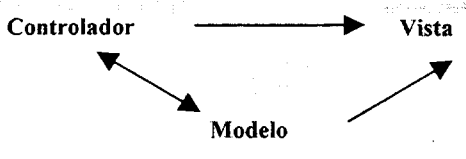


Figura 1 - MVC – Modelo/Vista/Controlador

Antes do MVC, os projetos de interface com o usuário tendiam a agrupar essas três funcionalidades do programa em um só objeto. O MVC separou esses objetos aumentando sua flexibilidade e permitindo uma melhor reutilização da estrutura. Uma vez separados, esses objetos comunicam aos outros quando sofrerem qualquer modificação, já que essas alterações podem interessar aos outros objetos que, dessa forma, poderão modificar seus parâmetros a fim de refletirem as modificações sofridas por cada um de seus pares. Essas idéias deram origem a um poderoso padrão de projeto, conhecido como *Observer*, onde um objeto de um programa (sujeito das alterações) notifica a todos os interessados (objetos observadores) que foi alterado e os observadores procuram adequar-se às modificações sofridas pelo objeto notificador,

atualizando-se. Desta forma, todas as visões do sistema refletem sempre os mesmos dados, mantendo-se coerentes entre si.

Padrões de projeto descrevem a forma com a qual os objetos podem comunicar-se sem a necessidade de acoplamentos, isto é, mantendo-se estruturalmente independentes (atributos e métodos). Manter essa separação e independência é um objetivo perseguido pelos bons programadores em orientação a objeto.

Os padrões de projeto começaram a ser reconhecidos, mais formalmente, no início dos anos 90 por Richard Helm (1990) e Erich Gamma (1992) e vieram a se consolidar em 1995 com a publicação do livro "Padrões de Projeto – Soluções Reutilizáveis de Software Orientado a Objetos". Este livro é comumente conhecido como "GoF" (*Gang of Four*) (Gamma et al, 1995), pelo fato de ter sido escrito por quatro autores de importância nessa área.

Os padrões de projeto são um caminho robusto para a reutilização de soluções de problemas que aparecem repetidamente em aplicações diversas. Sendo a reutilização de idéias um dos pontos fortes da orientação a objetos, é de fácil compreensão o crescimento intenso do interesse e da utilização dos padrões de projeto.

Durante os anos 90, a comunidade de engenharia de *software* reconheceu os benefícios da utilização de padrões de projeto e a sua aplicação foi rapidamente generalizada. Atualmente os padrões de projeto estão muito difundidos e, por conseqüência, bastante conhecidos pela comunidade de *software*. Todavia, sua aplicação não é trivial posto que existe um custo de complexidade embutido: saber utilizar corretamente os padrões de projeto em projetos de *software* um pouco mais elaborados requer um período de aprendizado significante por parte dos projetistas, que devem saber avaliar o correto uso desses padrões bem como a real necessidade da sua utilização naquela determinada aplicação.

Para aplicar um padrão de projeto o projetista precisa compreender bem a estrutura abstrata do padrão, seus benefícios e suas conseqüências, bem como as regras de comunicação entre seus objetos para em seguida adaptar esse padrão ao seu problema. Esse é um dos principais motivos da necessidade de uma ferramenta de ensino consistente e amigável capaz de auxiliar um aprendiz na tarefa de desenvolver sistemas orientados a objeto que incluam a utilização correta dos padrões.

Nosso trabalho de pesquisa – objeto do presente artigo – concerne ao projeto de implementação de um sistema dedicado ao ensino presencial ou remoto de padrões de projeto em sistemas orientados a objetos, acoplável a uma plataforma de EAD. A idéia base é oferecer uma ferramenta tecnológica de auxílio à escolha e ao uso dos padrões de projeto, considerando que o entusiasmo da comunidade de engenharia de *software* aponta para um grande crescimento da utilização desses padrões.

3. COMO E QUANDO APLICAR PADRÕES DE PROJETO

Os quatro elementos essenciais de um padrão de projeto, segundo (Gamma et al, 1995) são:

1. O **nome do padrão:** uma referência que deverá ser usada para associar a uma ou duas palavras, um problema de projeto, suas soluções e

conseqüências. A escolha apropriada do nome não parece ser uma tarefa fácil considerando que esse nome deve ser o mais sugestivo possível, a ponto de dar uma idéia do padrão em um número limitado de palavras. Dar nome a um padrão permite projetar em um nível mais alto de abstração, por ser possível utilizar um nome para referir-se a uma estrutura, por vezes complexa, de um padrão. Ter um vocabulário para padrões permite conversar sobre eles com outras pessoas, usá-lo em documentações e até mesmo proporcionar ajuda para se pensar no projeto com mais clareza.

2. O **problema:** descreve quando aplicar o padrão explicando o problema e seu contexto, bem como uma lista de condições que devem ser satisfeitas para que faça sentido aplicar o padrão. Problemas específicos de projeto, tais como representar algoritmos como objetos ou descrever estruturas de classe ou objeto, sintomáticas de um projeto inflexível, são alguns exemplos de caracterização do problema.

3. A **solução:** deve descrever os elementos que compõem o padrão de projeto, seus relacionamentos, suas responsabilidades e colaborações. A solução não deve descrever um projeto concreto ou uma implementação em particular, porque um padrão é como um gabarito que pode ser aplicado em muitas e diferentes situações. Em vez disso, o padrão deve fornecer uma descrição abstrata de um problema de projeto e de como um arranjo geral de elementos (classes, interfaces e objetos, no nosso caso) soluciona esse problema.

4. As **conseqüências:** devem ressaltar os resultados fazendo uma análise das vantagens e desvantagens da aplicação do padrão. Entendemos que a definição das conseqüências seja de grande importância para a avaliação de alternativas de projetos e para a compreensão dos custos computacionais e dos benefícios da aplicação do padrão. As conseqüências para o *software* freqüentemente envolvem compromissos de espaço de memória, tempo de processamento, e, em alguns casos, podem abordar aspectos sobre linguagens e implementações. Uma vez que a reutilização é considerada um fator importante no projeto orientado a objetos, as conseqüências de um padrão devem incluir o seu impacto sobre a flexibilidade e a portabilidade de um sistema.

Para usar um padrão de projeto, o profissional deve reconhecer as situações em que esse padrão se torna necessário.

Para reconhecer essas situações, o projetista certamente interromperá seu processo de criação a cada problema complexo encontrado e consultar o conjunto de padrões de projeto já formalizados, a fim de verificar se algum deles pode ajudá-lo a resolver completamente aquele determinado problema. Por outro lado, somente no catálogo de (Gamma et al, 1995) há vinte e três padrões de projeto formalizados.

Considerando o tamanho da descrição de cada padrão, o projetista teria que analisar uma quantidade considerável de informações na busca de condições para uma correta tomada de decisão. Muitas vezes torna-se difícil para um projetista identificar qual padrão deve ser utilizado na solução de seu problema, e, freqüentemente, não é possível resolvê-lo através da utilização de um padrão. O melhor caminho seria conhecer com profundidade o catálogo inteiro de padrões de projeto para reconhecer aquelas situações que requerem a utilização de algum deles. Isso não significa que o projetista tenha que ter na cabeça os detalhes de todos os padrões mais que ele deve ter um conhecimento consistente do campo de aplicação de cada um deles. Os padrões de projeto descrevem estratégias de projeto para resolver um problema geral, portanto eles não são imple-mentações que possam ser reutilizadas diretamente no código do programa.

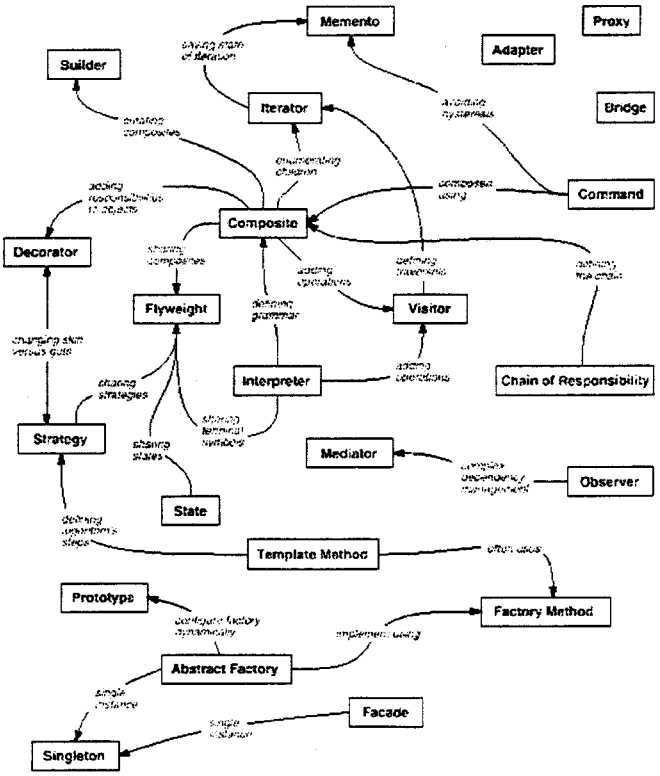


Figura 2 - Design Patterns e seus relacionamentos [GoFo95]

Para resolver um problema particular através de um padrão de projeto, o projetista precisa adaptar a estrutura abstrata do padrão à sua situação específica. Aplicar um padrão de projeto significa implementar esse padrão de acordo com um contexto particular: o contexto do seu problema.

Como podemos ver, aplicar um padrão de projeto não é uma tarefa trivial, nela o projetista precisa compreender bem a estrutura abstrata do padrão de projeto, seus benefícios e suas conseqüências, bem como as regras de comunicação entre seus objetos e então adaptar esse padrão ao seu problema. Desta forma, torna-se difícil para um iniciante em projetos envolvendo orientação a objetos valer-se dos benefícios da utilização de padrões de projeto sem o auxílio de uma ferramenta consistente e amigável que o ensine a utilizá-los corretamente.

4. O SISTEMA LeSOOP

O LeSOOP, proposto neste trabalho, está baseado por um lado em [Mote2000] no que concerne à apresentação dos padrões de projeto, suas conseqüências e seus relacionamentos, por outro lado, prevê a implementação de um sistema dedicado ao ensino remoto dos padrões de projeto em orientação a objetos, integrável a uma plataforma de EAD. O fato do sistema ser integrável a essa plataforma EAD não impede a sua utilização em aprendizado presencial, o que o torna mais versátil, ampliando as possibilidades de seu uso.

O protótipo desenvolvido em (Motelet, 2000) apresenta seis padrões. Nós estamos estendendo essa amplitude implementando os outros dezessete, que

compõem a coleção estudada e sugerida em (Gamma et al, 1995), abrangendo assim os vinte e três que compõem o catálogo, que estarão disponíveis no **LeSOOP**. A ferramenta em desenvolvimento dispõe de uma interface gráfica composta de uma janela principal que apresenta alguns menus onde se pode escolher o padrão a ser estudado (figura – 3).

No painel de cada padrão de projeto, são apresentadas as informações mais importantes para aprender sobre o respectivo padrão. Essas informações foram objeto de estudo de vários pesquisadores da área, inclusive dos autores de (Gamma et al, 1995):

- *Name*: o nome do padrão de projeto. Comunica sucintamente a sua essência.
- *Pourpose* : Descreve quando aplicar o padrão. Explica o problema e seu contexto.
- *Result*: a principal vantagem da utilização do padrão.
- *Variation*: um outro ponto de vista na intenção do padrão.
- *Mean*: através de que meios o padrão atinge seu objetivo.

Definir sobre quais princípios de projeto orientado a objetos este padrão está baseado é outra informação considerada por vários autores como de grande utilidade para o projetista. Os princípios destacados na ferramenta são:

- **Herança**, relacionamento que define uma entidade em termos de uma outra.
- **Composição**, montagem de objetos para obter um objeto composto com um comportamento mais complexo.
- **Interface**, conjunto de todas as assinaturas definidas pelas operações de um objeto. A interface descreve o conjunto de solicitações as quais um objeto pode responder. A interface não implementa esses métodos.
- **Classe abstrata**, classe cuja finalidade primária é definir uma interface. Uma classe abstrata delega às subclasses toda ou parte da tarefa da implementação, isto é, ela não pode ser instanciada.
- **Implementação direta**, classe que não tem operações abstratas, ou seja, classe instanciável.

O sistema **LeSOOP** apresenta o conjunto de princípios em que o padrão se baseia. Essa informação contém *links* (Figura 3.1) que levam o usuário a uma apresentação formalizada de cada um desses princípios.

Ao lado dos princípios, é apresentada a estrutura do padrão - representação gráfica das classes do padrão - usando uma notação baseada na Object Modeling Technique (OMT) (Rumbaugh et al, 1991) (Figura 3.2).

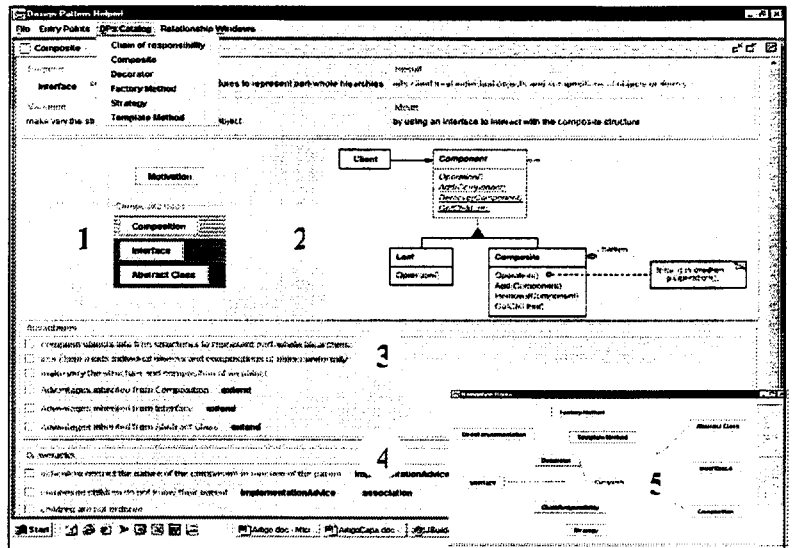


Figura 3 – Interface gráfica do sistema (apud Motelet, 2000)

Em outro painel, mais abaixo, são apresentadas, com riqueza de detalhes, as vantagens da aplicação do padrão e suas alternativas (Figura 3.3).

No último painel desse *frame*, são apresentadas, também com riqueza de detalhes, as desvantagens da aplicação do padrão, informações de implementação, alternativas, associações com outros padrões que funcionarão como antídotos dessa desvantagem etc. (Figura 3.4).

Outro ponto importantíssimo para a compreensão dos padrões é perceber os relacionamentos entre eles, suas diferenças e similaridades.

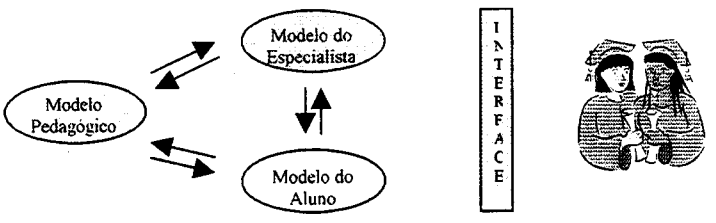
Tendo em vista a importância da compreensão desses relacionamentos, o sistema dispõe, em sua interface, de uma janela onde são sublinhadas as relações existentes entre os padrões, possibilitando ao usuário, em sua navegação, a consulta a esses padrões e/ou princípios relacionados (Figura 3.5).

O sistema tem também um conjunto de exemplos de aplicação dos padrões de projeto, implementados em Java, com explicações passo a passo de como os conceitos de cada padrão são utilizados para resolver o problema contextual, mostrando as vantagens e desvantagens dessa abordagem. Esses exemplos certamente servirão para clarear algumas dúvidas que, por ventura, permaneçam após a navegação através dos conceitos teóricos, facilitando a compreensão desses conceitos pelo projetista.

O LeSOOP também tem uma base de exercícios propostos e seus respectivos gabaritos. Durante o processo de aprendizagem, o usuário pode solicitar um exercício sobre o uso dos padrões. Aproveitando a tecnologia dos sistemas tutores inteligentes – ITS (Intelligent Tutoring System),o sistema escolhe um exercício em sua base e o disponibiliza para ser resolvido pelo estudante (no caso de não haver informações prévias sobre o nível de conhecimento desse aluno). A solução associada a esse exercício (gabarito) também é carregada, mas não apresentada ao aluno, com o objetivo de compará-la à solução desenvolvida pelo usuário. Um subsistema inteligente de diagnóstico tenta identificar o atual estágio de conhecimento do aluno durante a solução dos exercícios e o classifica, definindo seu perfil de conhecimento sobre os padrões de projeto. Depois disso, os exercícios propostos ao aluno serão escolhidos pelo sistema de acordo com o respectivo perfil estabelecido anteriormente.

Os ITS, em uma visão global, são sistemas tutoriais compostos de programas capazes de adaptar-se ao aluno, em oposição aos programas clássicos, onde o comportamento dos alunos é previsto em função de diferentes perfis pré-programados (Lima, 1992).

Algumas arquiteturas para construção desse tipo de sistemas foram propostas, mas a comunidade científica em ITS refere-se a uma arquitetura compreendida pelo menos dos seguintes três módulos: o modelo do especialista do domínio, o modelo pedagógico do sistema tutor e o modelo do aluno, como mostra a figura.



O modelo do especialista contém o máximo possível da representação do conhecimento a ser ensinado e é definido por especialistas daquele domínio. O modelo pedagógico contém as estratégias de ensino do conhecimento proposto, de acordo com o comportamento do aluno. Estes dois modelos são, na maioria dos casos, pré-definidos e permanecem imutáveis durante

a utilização do sistema. Já o modelo do aluno inicialmente está vazio, vai sendo definido durante a utilização do tutor, de acordo com o comportamento do aluno. Dessa forma o STI aprende as características de cada usuário, comportando-se de forma personalizada no processo de ensino do conteúdo do sistema. Acredita-se que esta estratégia facilita o aprendizado sobremaneira. A modelagem do aluno nos parece ser a atividade mais desafiadora deste projeto.

5. A PLATAFORMA

Estamos interagindo com um grupo de pesquisa em Engenharia de *Software,* no Núcleo de Computação Eletrônica – NCE, que vem desenvolvendo dois projetos aplicáveis na área de ensino a distância, nos quais o sistema LeSOOP será acoplado: o projeto Acádia e o projeto Dedalus.

A proposta do projeto Acádia é oferecer um ambiente de ensino-aprendizagem com interface gráfica que estimule a colaboração, aprendizagem e criatividade dos participantes de forma a ser o mais intuitivo possível, facilitando a interação homem-computador (Correa e Martins, 2002).

O sistema é composto de *web-pages* que simulam ambientes virtuais como Sala de Aula, Biblioteca e Secretaria, por exemplo. Outra meta do ambiente é aumentar a colaboração entre os alunos e entre o professor e seus alunos, abrindo espaço para discussão de dúvidas, sugestões e críticas. O projeto Acádia foi desenvolvido sobre uma plataforma gráfica chamada Dedalus (Paiva e Faissal, 2002) que utiliza a tecnologia "peer-to-peer" e recursos compartilhados. Ambos foram desenvolvidos utilizando os conceitos de orientação a objetos e em linguagem de programação Java, que possibilita a sua utilização em diferentes arquiteturas e sistemas operacionais. Além disso, vários padrões de projeto foram aplicados, a fim de obter um sistema flexível e expansível.

O sistema Dedalus é uma plataforma que pode ser usada como base por outras aplicações que envolvem tecnologia *peer-to-peer* e recursos compartilhados. Uma das aplicações da plataforma Dedalus é a utilização de ferramentas para o ensino à distância. A arquitetura proposta permite a esse tipo de aplicação uma vasta área de abrangência uma vez que permite a comunicação entre pessoas em pontos distintos da rede. Em termos de arquitetura, o fato de Dedalus ser federada traz benefícios quanto à performance, uma vez que os clientes poderão se comunicar independentemente de um servidor. Dedalus tem a grande vantagem de ser fácil de usar. Pessoas com pouco conhecimento tecnológico serão capazes de ter suas próprias páginas com componentes sofisticados de uma forma direta devido à interface amigável oferecida para o usuário.

6. CONCLUSÃO

Neste artigo, destacamos a importância dos padrões de projeto para o desenvolvimento de sistemas orientados a objeto como um passo importante na direção da geração de projetos de software reutilizáveis. Apresentamos o LeSOOP, um sistema para facilitar a compreensão e a aprendizagem dos conceitos visando a correta aplicação dos padrões de projeto, utilizando processos uniformes e contextuais de diagnóstico através de um banco de exercícios propostos, cujas soluções desenvolvidas pelos estudantes servem de base para a especificação de

seus níveis de conhecimento. Ressaltamos também a importância de acoplar a ferramenta LeSOOP no sistema Acádia e na plataforma Dedalus, desenvolvidos para aplicações de ensino a distância, utilizando a tecnologia "peer-to-peer" e interfaces amigáveis, capazes de guiar o usuário durante a utilização desses sistemas, evitando que eles se sintam perdidos durante o processo de ensino/aprendizagem.

7. BIBLIOGRAFIA

Alexander, C. **A pattern language**. New York Oxford University Press, 1977.

Cooper, J. **The Design Patterns: Java Companion**. Addison Wesley Design Patterns Series, 1988.

Correa, B. e Martins C. **Ambiente de ensino-aprendizagem com ênfase em comunicação gráfica, desenvolvido em plataforma distribuída** Núcleo de Computação Eletrônica / Universidade Federal do Rio de Janeiro – Brasil, 2002.

Gamma, E.; Helm, R.; Johnson, R. e Vlissides, J. **Design Patterns. Elements of Reusable Software** Addison-Wesley, Reading, MA, 1995.

Krasner, G.E. e Pope, S.T. **A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80** Journal of Object-Oriented Programmng I(3)., 1988.

Lima, C. **Sintonia: Vers um Système Intelligent pour la Construction des Diagnostics dans les Tuteurs d'Informatique** Thèse de Doctorat, Université Paris VI, 1992.

Michalski, R. S., Carbonell, J. G. e Mitchell, T. M. **Machine Learning: An Artificial Intelligence Approach** Morgan Kaufmann Publishers, Inc.; pp 26-27, 1983.

Motelet, O. **A contextual help system for assisting OO software designers in using design patterns** Tese de mestrado. Vrije Universiteit Brussels – Bélgica / École des Mines de Nantes – França / Universidade do Brasil – UFRJ, 2000.

Paiva, D. e Faissal, G. **Uma Plataforma Java para Aplicações Peer-to-Peer** Núcleo de Computação Eletrônica / Universidade Federal do Rio de Janeiro – Brasil, 2002.

Romanczuck-Requile, A., Lima, C., Kaestner, C., e Scalabrin, E. ; **A contextual help system based on intelligent diagnosis processing aiming to design and maintain object-oriented package**; Lecture Notes in Computer Science; issn 0302-9743; Springer-Verlag, pp. 64-65, 1988.

Rumbaugh, I.; Blaha, M.; Premerlani, W.; Eddy, F. e Lorenson, W. **Object-Oriented Modeling and Design**. Prentice Hall, Englewood Cliffs, NJ, 1991.